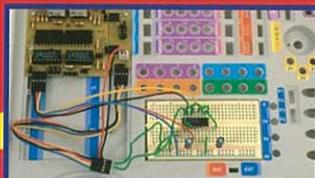
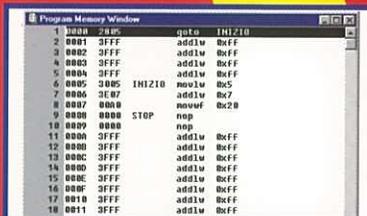
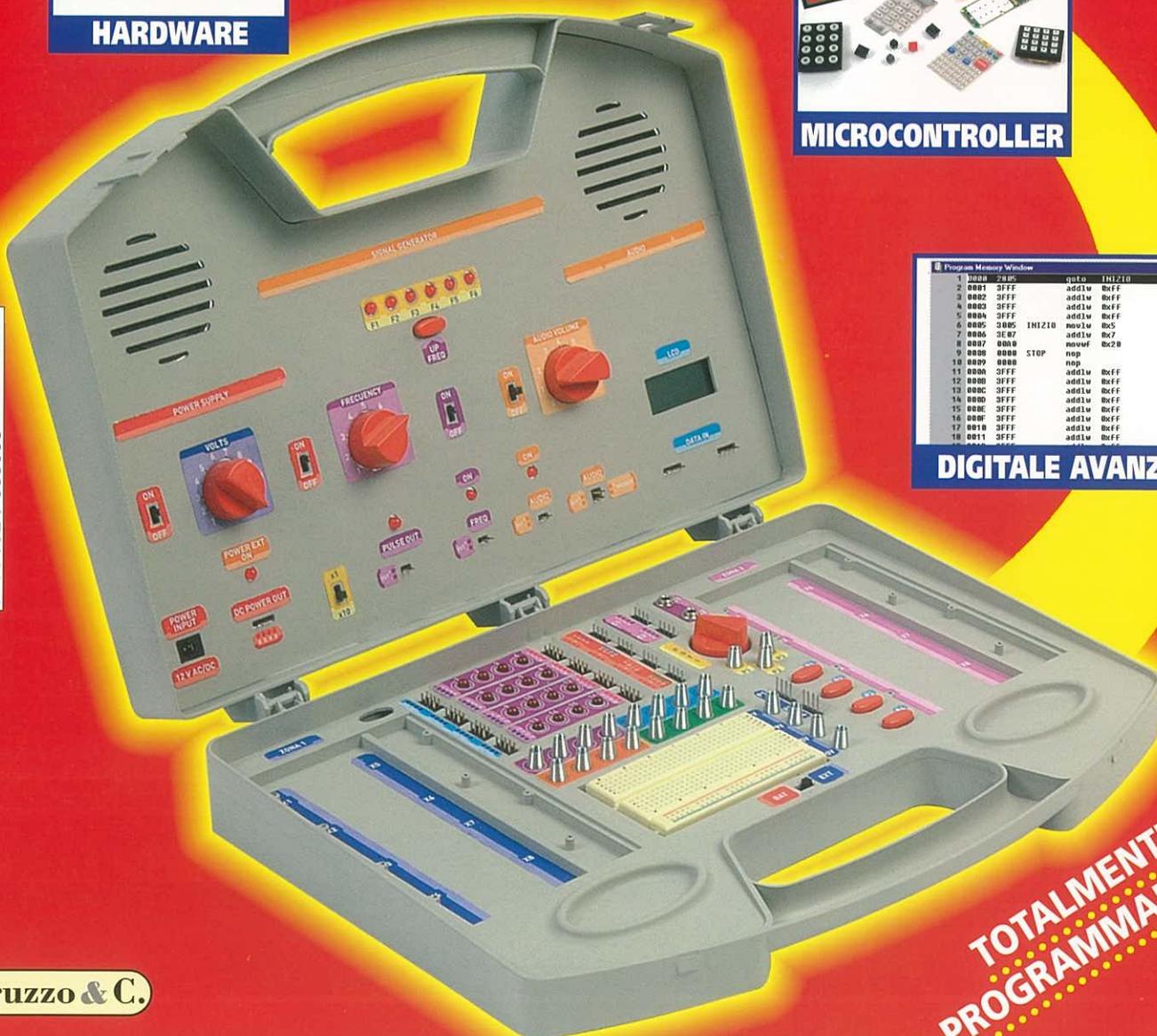
## ...e costruisci il tuo LABORATORIO DIGITALE

6,90 €

16

HARDWARE

DIGITALE DI BASE

MICROCONTROLLER

DIGITALE AVANZATO

TOTALMENTE PROGRAMMABILE!!!

9 771824 363008 41016

Peruzzo & C.

**Direttore responsabile:**
ALBERTO PERUZZO
**Direttore Grandi Opere:**
GIORGIO VERCELLINI
**Consulenza tecnica e traduzioni:**
CONSULCOMP S.n.c.
**Pianificazione tecnica**
LEONARDO PITTON

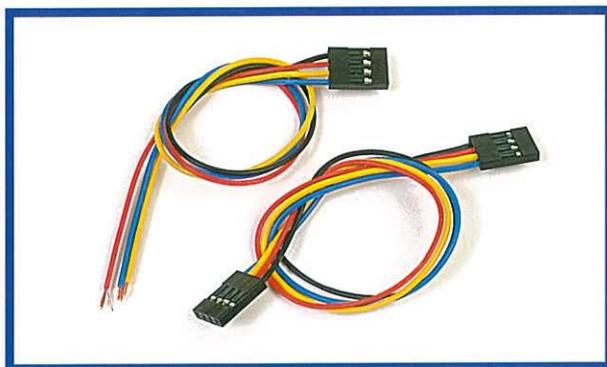
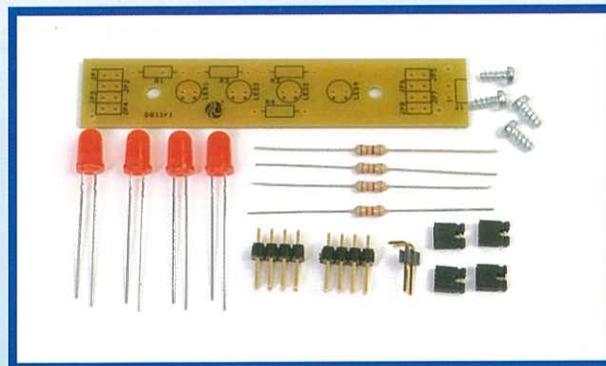
Direzione, Redazione, Amministrazione: viale Ercole Marelli 165, Tel. 02/242021, 20099 Sesto San Giovanni (Mi). Pubblicazione settimanale. Registrazione del Tribunale di Monza n. 1738 del 26/05/2004. Spedizione in abbonamento postale gr. II/70; autorizzazione delle Poste di Milano n. 163464 del 13/2/1963. Stampa: Staroffset s.r.l., Cernusco S/N (MI). Distribuzione SO.DI.P. S.p.A., Cinisello Balsamo (MI).



"ELETTRONICA DIGITALE" si compone di 70 fascicoli settimanali da suddividere in 2 raccoglitori.

**RICHIESTA DI NUMERI ARRETRATI.** Per ulteriori informazioni, telefonare dal lunedì al venerdì ore 9.30-12.30 all'ufficio arretrati tel. 02/242021. Se vi mancano dei fascicoli o dei raccoglitori per completare l'opera, e non li trovate presso il vostro edicolante, potrete riceverli a domicilio rivolgendovi direttamente alla casa editrice. Basterà compilare e spedire un bollettino di conto corrente postale a PERUZZO & C. s.r.l., Ufficio Arretrati, viale Marelli 165, 20099 Sesto San Giovanni (MI). Il nostro numero di c/c postale è 42980201. L'importo da versare sarà pari al prezzo dei fascicoli o dei raccoglitori richiesti, più le spese di spedizione € 3,10 per pacco. Qualora il numero dei fascicoli o dei raccoglitori sia tale da superare il prezzo globale di € 25,82 e non superiore a € 51,65, l'invio avverrà per pacco assicurato e le spese di spedizione ammonteranno a € 6,20. La spesa sarà di € 9,81 da € 51,65 a € 103,29; di € 12,39 da € 103,29 a € 154,94; di € 14,98 da € 154,94 a € 206,58; di € 16,53 da € 206,58 in su. Attenzione: ai fascicoli arretrati, trascorse dodici settimane dalla loro distribuzione in edicola, viene applicato un sovrapprezzo di € 0,52, che andrà pertanto aggiunto all'importo da pagare. Non vengono effettuate spedizioni contrassegno. Gli arretrati di fascicoli e raccoglitori saranno disponibili per un anno dal completamento dell'opera. IMPORTANTE: è assolutamente necessario specificare sul bollettino di c/c postale, nello spazio riservato alla causale del versamento, il titolo dell'opera nonché il numero dei fascicoli e dei raccoglitori che volete ricevere.

## IN REGALO in questo fascicolo

1 Cavetto a quattro fili con due connettori femmina a quattro contatti
1 Cavetto a quattro fili con un connettore femmina a quattro contatti

## IN REGALO nel prossimo fascicolo

1 Scheda DG11R1
4 LED rossi
2 Connettori maschio diritti 2x4
1 Connettore maschio a 90° 2x1
4 Resistenze da 820 Ω 5% 1/4 W
4 Ponticelli isolati neri
4 Viti

## COME RACCOGLIERE E SUDDIVIDERE L'OPERA NELLE 4 SEZIONI

L'Opera è composta da 4 sezioni identificabili dalle fasce colorate, come indicato sotto. Le schede di ciascun fascicolo andranno suddivise nelle sezioni indicate e raccolte nell'apposito raccoglitore, che troverai presto in edicola. Per il momento, ti consigliamo di suddividere le sezioni in altrettante cartellette, in attesa di poterle collocare nel raccoglitore. A prima vista, alcuni numeri di pagina ti potranno sembrare ripetuti o sbagliati. Non è così: ciascuno fa parte di sezioni differenti e rispecchia l'ordine secondo cui raccogliere le schede.
**Per eventuali domande di tipo tecnico scrivere al seguente indirizzo e-mail:**
elettronicadigitale@microrobots.it

**Hardware** Montaggio e prove del laboratorio

**Digitale di base** Esercizi con i circuiti digitali

**Digitale avanzato** Simulazione con MPLAB (I)

**Microcontroller** Esercizi con i microcontroller

# Cavetti

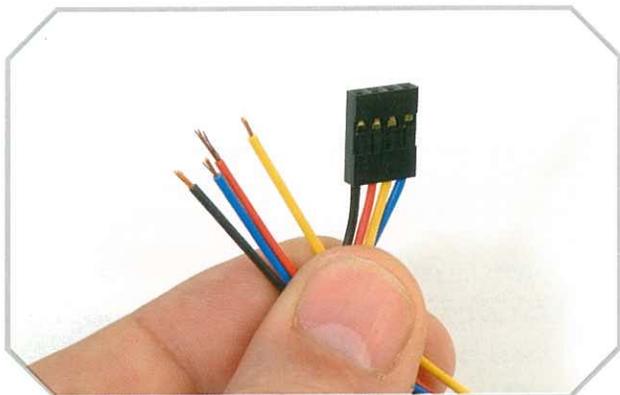
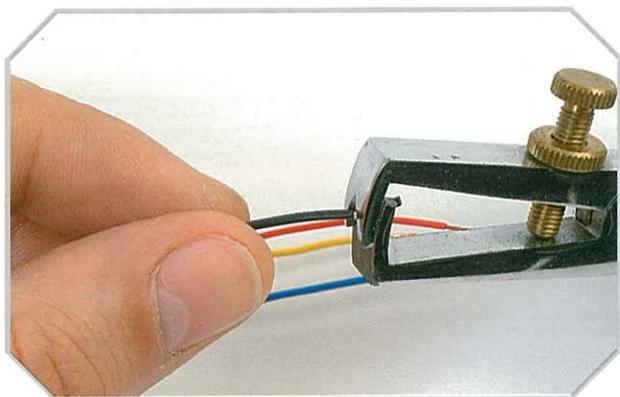
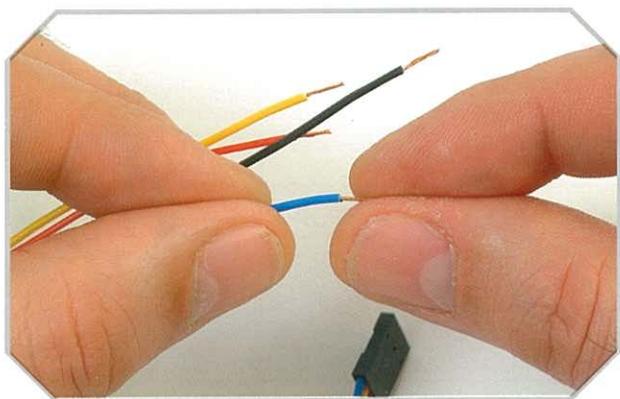
*Cavetti di collegamento.*

***Inizia la fornitura dei cavetti di collegamento partendo con due di essi a quattro conduttori: un modello termina con due connettori, l'altro a una estremità ha un connettore e all'altra fili indipendenti e spelati. Più avanti vi verranno forniti altri cavetti di questo tipo, che saranno molto utili, perché permettono il collegamento immediato tra i diversi circuiti o gli elementi ausiliari del laboratorio.***

*Cavetto con un connettore volante e fili spelati all'altro capo.*

## Idea di base

Nelle prove di laboratorio è molto frequente impiegare parecchio tempo nel cablaggio degli esercizi, quindi, allo scopo di velocizzare questa operazione, alcune schede del laboratorio dispongono di terminali di collegamento a torretta, in posizione verticale o orizzontale, raggruppati preferibilmente di 4 in 4 per permettere il collegamento simultaneo di 4 terminali. Questo tipo di collegamento è utilizzato anche per le matrici dei diodi LED, i pulsanti e altri circuiti del laboratorio.

*Cavetto terminato con due connettori volanti a quattro conduttori.*

*Dettaglio del connettore.*

*È necessario togliere circa 6 mm di copertura isolante dal filo.*

*I fili vanno ritorti per fare in modo che rimangano raggruppati.*

Oltre a risparmiare tempo si ottiene anche un buon collegamento, sia dal punto di vista elettrico che meccanico, e rende più chiaro il montaggio, visto che, quando si utilizzano i fili indipendenti, alcuni circuiti complessi possono sembrare la tela di un ragno.

## Collegamenti

Il collegamento sul cavetto terminato con due connettori è, come si dice in gergo, "pin to pin", ovvero terminale 1 con terminale 1 ottenuto attraverso il filo nero, terminale 2 con terminale 2 ottenuto attraverso il filo rosso, terminale 3 con terminale 3 ottenuto attraverso il filo giallo, e, infine, terminale 4 con terminale 4 ottenuto attraverso il filo azzurro.

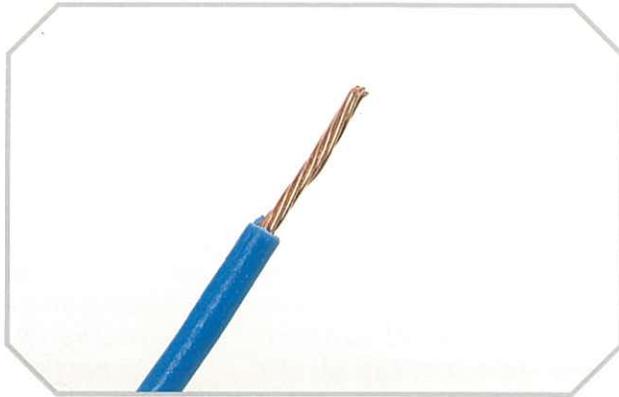
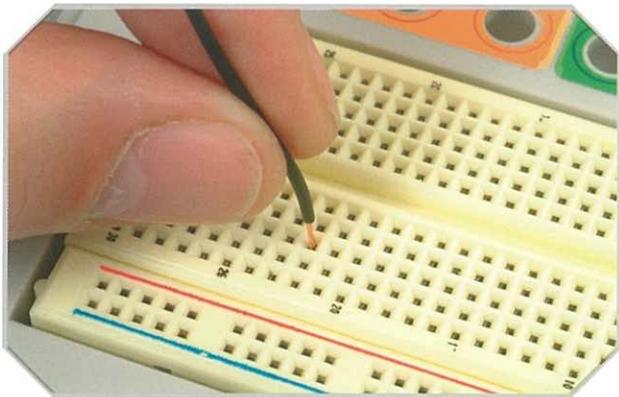
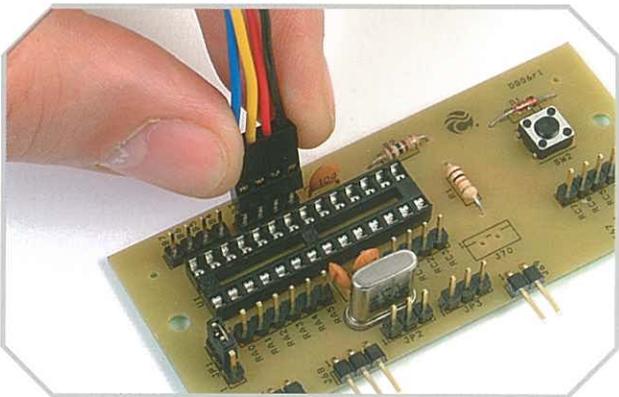
## I connettori

I connettori utilizzati nei cavetti sono di tipo femmina, con una separazione standard tra i terminali di 2,54 mm. L'elemento fondamentale del connettore consiste in un terminale femmina che viene fornito già collegato mediante pressione su ogni filo, operazione per la quale è necessario disporre di un attrezzo adeguato per garantire il collegamento. L'altra parte del connettore consiste in un elemento in plastica nera che contiene i vari contatti, nel nostro caso quattro. Su questo elemento troviamo modellata una piccola ritenzione dove si fisserà una minuta linguetta metallica del terminale femmina che, una volta inserita, non si potrà più estrarre, dato che si aggancerà in maniera simile a un arpione.

Ogni connettore viene fornito completamente montato, e questo vale anche per i quattro terminali montati sulla scheda.

Questo tipo di connettore è pensato per essere collegato e scollegato molte volte durante i diversi esercizi che realizzeremo, è privo di sistemi di fissaggio sulla scheda, ovvero si può collegare e scollegare tirando il connettore stesso, non i fili, senza la necessità di realizzare nessun'altra operazione addizionale.

## I fili

In questi cavetti viene utilizzato del filo flessibile da 0,5 mm$^2$ di sezione, di quattro diversi colori, distribuiti nei seguenti modi: nero per il filo 1, rosso per il filo 2, giallo per il filo 3 e azzurro per il filo 4.

*Aspetto finale del filo.*

*Inserimento dell'estremo del filo nella scheda Bread Board.*

*I terminali delle schede sono predisposti per l'utilizzo di cavetti di interconnessione.*

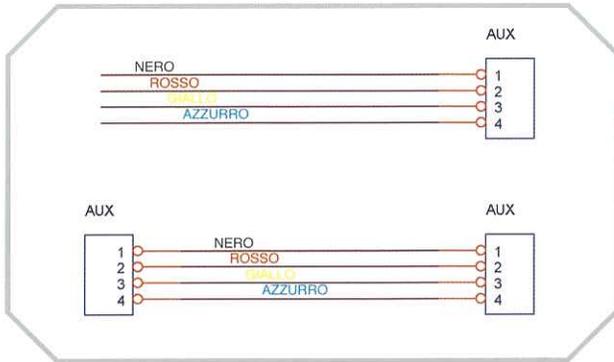
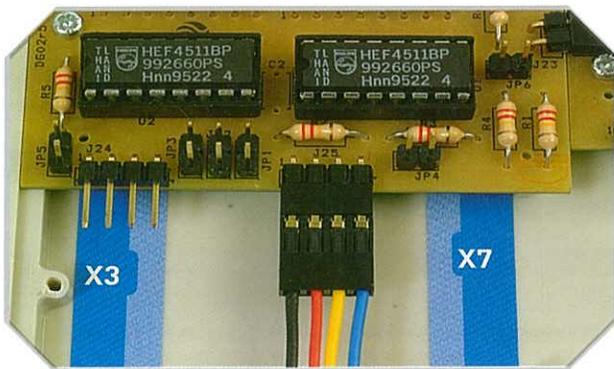
Fondamentalmente vi sono due tipi di cavetti, uno terminato su due connettori e l'altro con gli estremi di ogni filo spelati per circa 6 mm. Questa parte di filo spelato può essere usata per collegarlo alle molle a pressione che ancora non vi sono state fornite o alla scheda Bread Board, tuttavia, per quest'ultimo utilizzo, bisogna verificare che i reofori siano tutti uniti e leggermente ritorti, in modo da rimanere raggruppati, per poter essere inseriti con facilità nei fori di collegamento della Bread Board. I fili sono stati spelati per 6 mm, però, dato che l'isolamento non è incollato al filo allo scopo di poterne togliere una parte quando è necessario, potrebbe scivolare sul filo e la lunghezza della parte spelata potrebbe variare durante il trasporto. Normalmente basta tirare leggermente il conduttore per farlo fuoriuscire di 6 mm. Se non fosse così, è possibile asportare la parte di isolamento necessaria, e a questo scopo vi consigliamo di utilizzare uno spelafili per non danneggiare nessuno dei reofori di rame.

## Disposizione

Questo tipo di connettore non ha polarità, tuttavia conviene seguire sempre lo stesso ordine per evitare di commettere errori, utilizzando sempre il nero per collegarlo al terminale 1 del connettore in cui si inserisce il cavetto.

Il tipo di connettore maschio utilizzato è quello aperto, che permette di collegare un connettore femmina del cavetto su un connettore maschio da 2 o 3 terminali; in questo caso è evidente che si utilizzeranno solamente i cavi collegati. Tutto ciò è possibile se non esistono elementi nelle vicinanze che blocchino meccanicamente l'operazione.

## Utilizzo

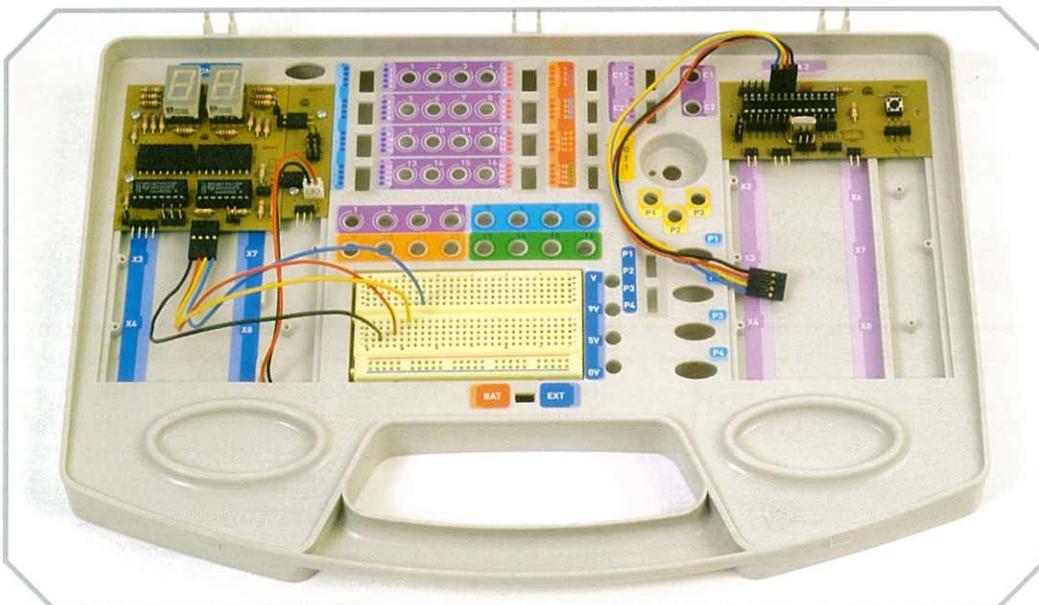
Per fare in modo che questi cavetti mantengano le loro caratteristiche devono essere utilizzati solo con i connettori maschio corrispondenti e, in questo caso, con fili da 0,5 mm flessibili o rigidi. È necessario evitare usi diversi che potrebbero danneggiarli, come gli strattoni forti per scollegarli. Il collegamento si deve realizzare con i connettori maschio e femmina ben allineati per evitare la deformazione meccanica e il deterioramento prematuro degli stessi.

*Schema elettrico dei cavetti.*

*Il filo nero si deve sempre collegare al terminale 1 del connettore.*

## Manutenzione

Con un uso normale e frequente, la parte spelata del filo si deteriora a causa dell'attorcigliamento e del logorio, quindi è consigliabile risanare il capo del filo tagliando la parte danneggiata e asportando una piccola parte di isolante. Se, per qualche motivo, uno dei contatti fuoriesce dal connettore, è possibile, se questo non è rotto, tornare a riposizionarlo all'interno del supporto nero osservando come sono montati gli altri, facendo particolare attenzione alla piccola molla che impedisce la sua uscita quando si collega o si tira il cavo. È possibile che questa molla si sia deformata, sarà quindi necessario conformarla nuovamente con piccole pinzette a punta sottile, affinché torni a compiere la sua funzione di incastro.

## Schema

Per facilitare il montaggio abbiamo rappresentato questi cavetti nello schema, identificando i connettori come AUX seguito da un numero, nel caso se ne utilizzi più di uno, per poter distinguere gli uni dagli altri.

Inoltre, per facilitare i lavori del montaggio pratico, viene anche indicato a quale connettore di quale scheda deve essere collegato, quindi è necessario seguire e utilizzare lo schema e le illustrazioni come aiuto.

*L'utilizzo dei cavetti permette di realizzare rapidamente i collegamenti.*

# Esperimenti con ritardi

***In questo esperimento si utilizza un circuito generatore la cui uscita è applicata a due porte invertenti, collegate in modo che l'uscita della prima sia applicata all'ingresso della seconda. Il segnale di uscita del generatore e quello di ogni porta invertente arrivano all'ingresso di uno dei codificatori BCD-7 segmenti del circuito stampato DG02. Il condensatore C2 si utilizza per generare un ritardo.***

## Idea di base

Tra le limitazioni di velocità dei circuiti digitali vi sono le capacità parassite, che possono ritardare un segnale e farlo giungere fuori tempo, inserendo nel circuito di ricezione un dato sbagliato.

Nel nostro caso e per adattarci al materiale disponibile, montiamo un piccolo circuito che lavora a una velocità molto bassa, per fare in modo che l'esperimento sia facile da osservare. Inseriamo una capacità elevata, in questo caso C2, per generare un ritardo. L'effetto che questa alta capacità genera in questo circuito di frequenza molto bassa è paragonabile a quello che può provocare un condensatore di pochi picofarad in un circuito che lavora ad alta frequenza.

## Lo schema

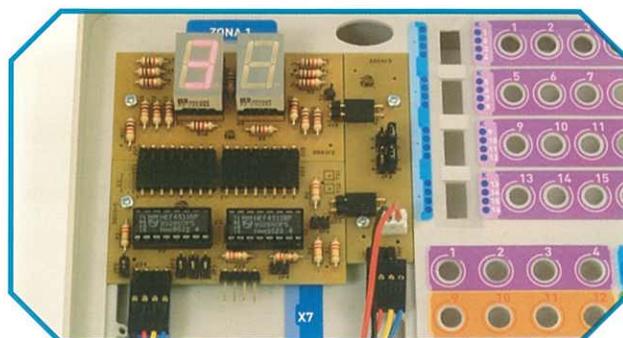
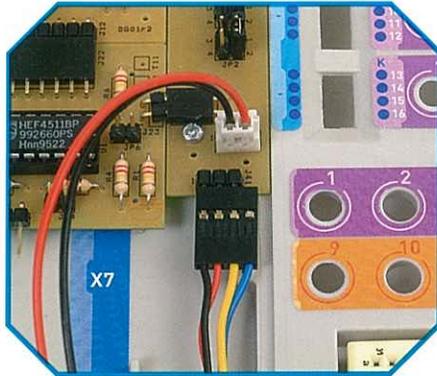
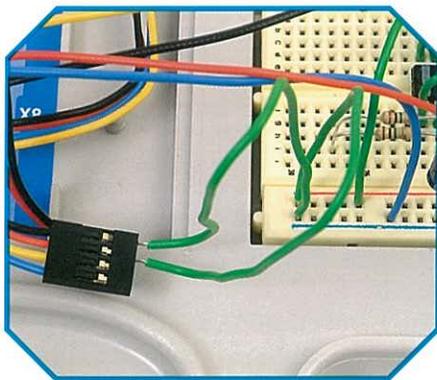
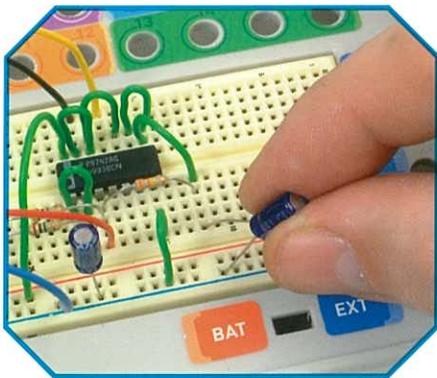
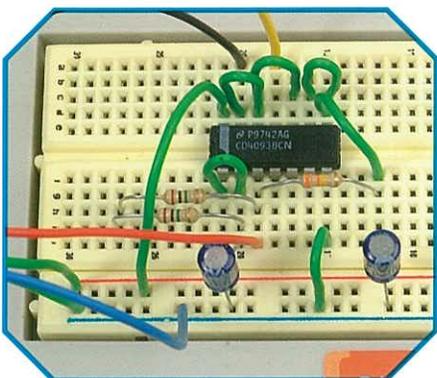
Il circuito utilizza quattro porte NAND dell'integrato 4093. La porta U1A si utilizza per costruire un oscillatore astabile di frequenza molto bassa, e la sua uscita, il terminale 3 dell'integrato, cambia da 0 a 1. Supponiamo per un momento che il condensatore C2 non esista. L'uscita di U1A si applica all'ingresso di U3C che funziona come una porta invertente,

*In questo esperimento si utilizzano i circuiti stampati DG01, DG02 e DG04.*

dato che i suoi ingressi sono uniti tra loro. Sul terminale 10 il segnale sarà invertito rispetto all'ingresso. L'uscita di questo invertitore si porta all'ingresso della porta invertente successiva costruita con la porta U4D, il segnale di uscita si ottiene sul terminale 11 e deve essere uguale a quella del terminale 3.

## Montaggio

Il montaggio si realizza sulla scheda Bread Board tenendo conto della polarità dei condensatori elettrolitici. Bisogna anche fare

*Circuito per la simulazione di un ritardo dovuto a una capacità.*

*Il connettore del cavetto si collega al connettore J24 del circuito stampato DG02, in modo che al filo nero corrisponda sempre il terminale 1 del connettore.*

*Il cavetto con due connettori volanti alle sue estremità si utilizza per portare l'alimentazione dal connettore J41 della scheda DG04 fino alla scheda Bread Board.*

*Collegamenti dell'alimentazione alla scheda Bread Board.*

*Il condensatore C2 si utilizza per generare un ritardo nella trasmissione del segnale.*

*Scheda Bread Board con l'esperimento montato.*

molta attenzione all'orientamento del circuito integrato. Utilizziamo il cavetto di collegamento inserendo il suo connettore femmina da 4 terminali e quattro fili sciolti, identificato come AUX3 nello schema, su J24 del circuito stampato DG02, collegando il nero al terminale 1 dello stesso. Collegheremo poi l'altro estremo del cavo nero a uno qualsiasi dei terminali 10, 12 o 13 dell'integrato, dato che sono uniti tra loro.

Il cavo rosso si collega al terminale 3 del circuito integrato, quello giallo al terminale 11 e l'azzurro direttamente al negativo dell'alimentazione.

## Alimentazione

Questo circuito si può alimentare utilizzando il cavetto di collegamento che ha due connettori alle sue estremità, uno dei quali si collegherà al connettore J41 del circuito stampato DG04, in modo che il filo nero sia sempre collegato al terminale 1 del connettore. All'altra estremità l'alimentazione si collegherà con due cavetti, rosso al positivo e nero al negativo, mentre i ponticelli della scheda DG04 devono essere inseriti sui terminali 1 e 2.

## Funzionamento

Anche se il segnale sui terminali 3 e 11 dell'integrato è lo stesso, è necessario tener conto che, dal momento in cui esce dal terminale 3 fino a quando arriva al terminale 11, impiega un determinato tempo, normalmente molto breve e non apprezzabile per alcune applicazioni, tutto dipende dal momento in cui è necessario il dato. Collegando l'alimentazione e senza inserire C2, sul display possiamo leggere alternativamente i numeri 1 e 6; se provochiamo un ritardo collegando il condensatore C2, vedremo visualizzati per un breve periodo di tempo i valori 3 e 4.
Con i valori indicati nello schema, si può vedere molto bene l'effetto, se diminuiamo il ritardo abbassando, per esempio, la resistenza R3 a 47 K, l'effetto si fa minore anche se continua a essere visibile.

**LISTA DEI COMPONENTI**

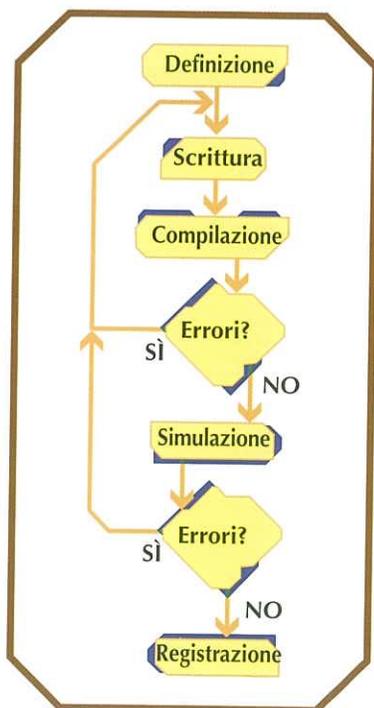
| Circuito base | |
|---|---|
| U1 | Circuito integrato 4093 |
| R1, R2 | Resistenza 1M (marrone, nero, verde) |
| R3 | Resistenza 330K (arancio, arancio, giallo) |
| C1, C2 | Condensatore 10 µF elettrolitico |

# Simulazione con MPLAB (I)

***Nei capitoli precedenti abbiamo completato un programma e lo abbiamo liberato dagli errori: questo ha generato un file con estensione ".hex" che verrà scritto sul microcontroller, però rimangono ancora alcune cose importanti da fare. Il programmatore, prima di scrivere il programma sul PIC, deve assicurarsi che il codice creato corrisponda ai requisiti del progetto, ovvero che il suo programma faccia ciò che realmente si desidera. MPLAB offre la possibilità di simulare il programma, il suo potente simulatore ci permetterà di analizzare come funzionerebbe il programma se stesse girando su un PIC.***

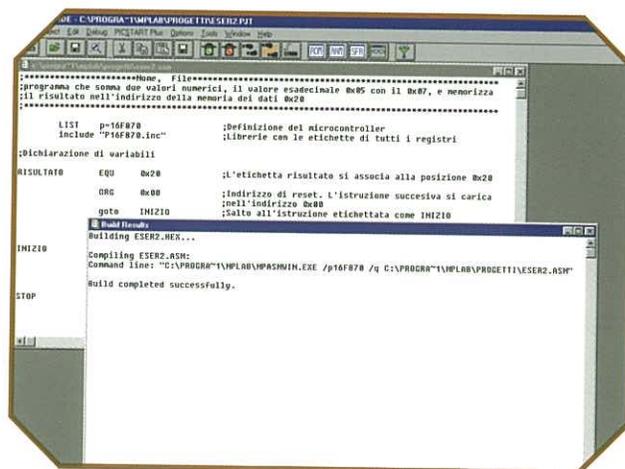
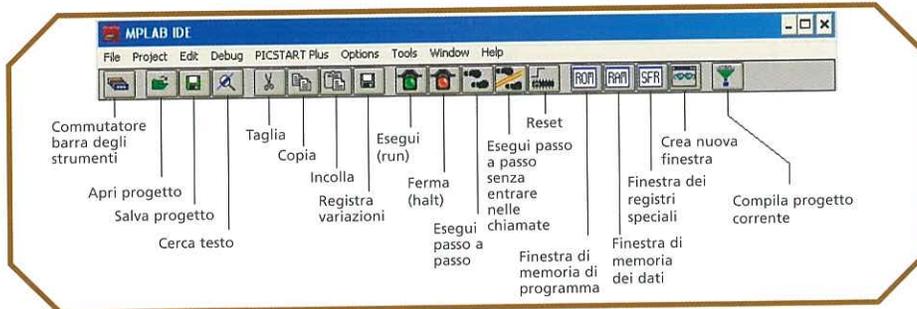
## Prima di simulare

Dobbiamo aprire MPLAB e caricare il progetto con cui stiamo lavorando. Dopo averlo aperto lo compileremo nuovamente per assicurarci che il nostro programma non contenga errori. Apparirà una videata per indicarci che la compilazione si è realizzata con successo, ovvero è stato creato un file con estensione ".hex" che può già essere scritto sul microcontroller.

Prima di scrivere il programma sul microcontroller è conveniente verificare che corrisponda realmente ai requisiti dell'applicazione, utilizzando il simulatore.

*Fasi del progetto di un programma.*

## La barra degli strumenti di simulazione

Se commutiamo tra le diverse barre degli strumenti, arriveremo a una identica, quella della figura alla pagina successiva. Benché ogni barra abbia una sua utilità specifica, possiamo vedere che alcune delle funzioni si ripetono sulle varie barre. Questo si fa per non dover sempre commutare tra le barre ogni volta che avremo bisogno di eseguire alcune azioni che esulano dall'utilità specifica propria della barra, ovvero, in ogni barra avremo le funzioni specifiche della barra stessa più alcune funzioni generali, in modo da semplificare la gestione del software.

Nella barra di simulazione, quindi, potremo vedere che le prime otto funzioni sono generali e le successive proprie della simulazione. L'ultima delle funzioni è quella di compilazione, dato che nella fase di simulazione potre-

*Videata dove viene indicato il risultato della compilazione.*

*Barra degli strumenti di simulazione.*

mo modificare il programma, però prima di simularlo nuovamente sarà necessario tornare a compilarlo.

## Finestre o window

Le finestre permettono di vedere cosa fa il nostro programma, come funziona internamente al PIC mano a mano che si sta eseguendo. Quando attiviamo l'opzione window, nella barra appare il menù a tendina della figura sottostante.

*Menù a tendina dell'opzione window della barra dei menù.*

## Program memory

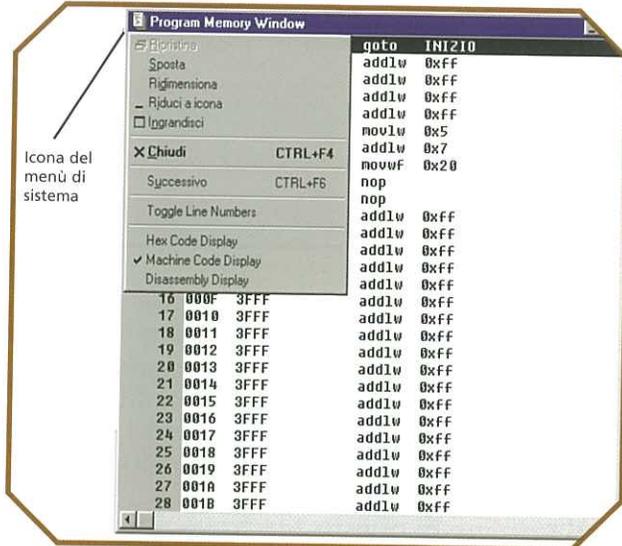
Selezionando questa opzione appare la videata riportata nella figura sottostante in cui è possibile vedere le posizioni della memoria occupate da ogni istruzione, il codice delle operazioni di ogni istruzione e la posizione di memoria che è stata assegnata a ogni etichetta.

Se clicchiamo con il mouse sull'icona nell'angolo superiore sinistro di questa finestra, la barra di menù del sistema, appare il menù a tendina della figura in cui potremo selezionare tre modi di vedere la memoria di programma:

Hex Code Display: rappresenta la memoria di programma con i dati in formato esadecimale. Questa opzione è utile quando vogliamo verificare se i dati sul microcontroller sono stati scritti correttamente, confrontando la videata del software di scrittura utilizzato se abbiamo scelto di scrivere in questo formato.

Program Memory Window

```
 1 0000  2805          goto   INICIO
 2 0001  3FFF          addlw  0xff
 3 0002  3FFF          addlw  0xff
 4 0003  3FFF          addlw  0xff
 5 0004  3FFF          addlw  0xff
 6 0005  3005  INIZIO  movlw  0x5
 7 0006  3E07          addlw  0x7
 8 0007  00A0          movwf  0x20
 9 0008  0000  STOP    nop
10 0009  0000          nop
11 000A  3FFF          addlw  0xff
12 000B  3FFF          addlw  0xff
13 000C  3FFF          addlw  0xff
14 000D  3FFF          addlw  0xff
15 000E  3FFF          addlw  0xff
16 000F  3FFF          addlw  0xff
17 0010  3FFF          addlw  0xff
18 0011  3FFF          addlw  0xff
19 0012  3FFF          addlw  0xff
20 0013  3FFF          addlw  0xff
21 0014  3FFF          addlw  0xff
22 0015  3FFF          addlw  0xff
23 0016  3FFF          addlw  0xff
24 0017  3FFF          addlw  0xff
25 0018  3FFF          addlw  0xff
26 0019  3FFF          addlw  0xff
27 001A  3FFF          addlw  0xff
28 001B  3FFF          addlw  0xff
```

*Videata della finestra di memoria di programma.*

*Apertura delle opzioni del menù di sistema.*

*Machine Code Display:* questa opzione è quella predeterminata nel software e presenta il codice macchina assemblato così come abbiamo visto in precedenza.

*Disassembly Display:* espande il codice esadecimale disassemblandolo con i simboli, quindi risulta più facile seguire il programma.

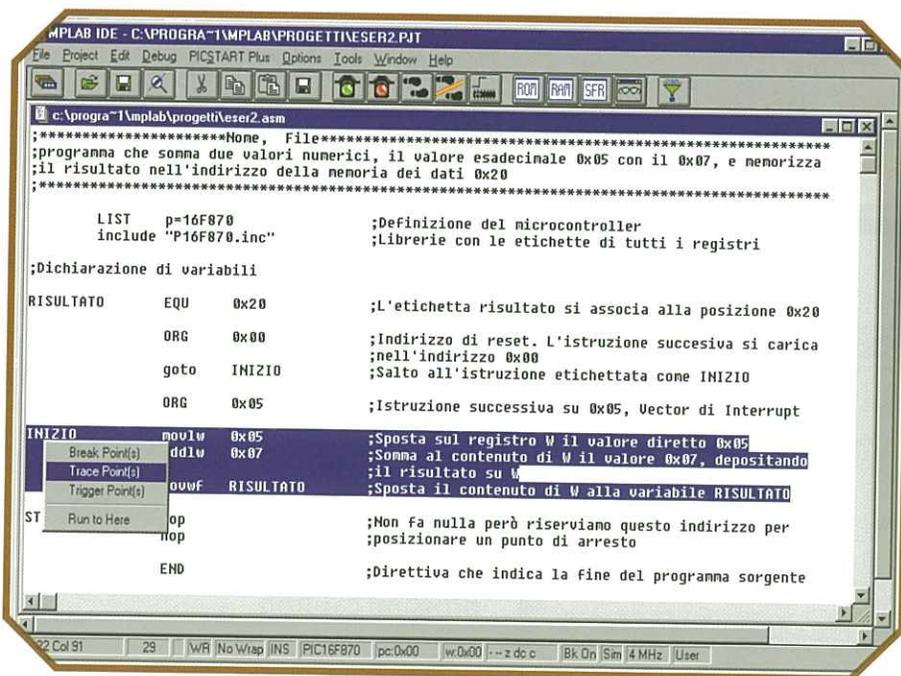
Nei due modi precedenti, Machine Code e Disassembly, l'istruzione a cui punta il contatore di programma si trova evidenziata.

## Trace memory

Questa finestra si chiama traccia di memoria e la sua funzione è quella di catturare in un preciso momento l'evoluzione del programma quando questo sta girando in tempo reale. Alcuni problemi si evidenziano solamente quando il programma sta girando e non durante l'esecuzione passo a passo, quindi, per rilevarli, utilizzeremo questo strumento. Mediante

*Traccia di memoria ottenuta con le linee selezionate.*

questa finestra potremo visualizzare un registro durante l'esecuzione del programma, registrando gli stati che assume per una successiva analisi. Prima di attivare questa opzione, evidenzieremo con il mouse le linee di codice delle quali vogliamo ottenere i dati durante l'esecuzione del programma e cliccheremo il pulsante destro del mouse. Apparirà il menù a tendina della figura.

Se selezioniamo Trace Point le linee selezionate verranno evidenziate in colore verde. Per eseguire il programma selezioneremo il semaforo verde dalla barra degli strumenti o sul menù di controllo Debug→Run→Run. Se dopo qualche secondo (il simulatore emula il funzionamento del PIC, ma è molto più lento di quest'ultimo) attiveremo l'icona del semaforo rosso oppure sul menù di controllo Debug→Run→Halt, fermeremo l'ese-

*Selezione delle linee di programma da caricare nella finestra di tracciamento.*

EEPROM Window

| ---- | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF |
| 0010 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF |
| 0020 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF |
| 0030 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF |

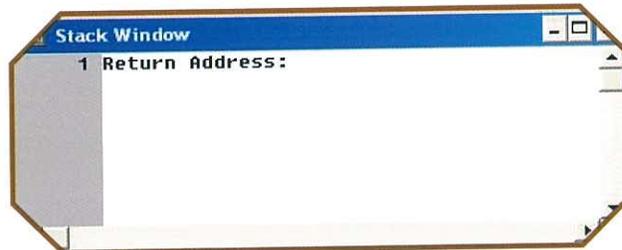
*Finestra della memoria EEPROM.*

cuzione del programma. Se ora attiveremo l'opzione Trace Memory potremo vedere la traccia ottenuta. Nella figura si può vedere la traccia ottenuta per il nostro esempio. Il simulatore mostra in questa finestra il valore del tempo impiegato nell'eseguire ogni linea di programma, oltre che qualsiasi variazione prodotta sui registri dall'esecuzione del codice di istruzione.

## EEPROM Memory

Questa finestra è utile solamente per i PIC che dispongono di tale dispositivo. Il contenuto di questa memoria si può vedere selezionando l'opzione Window→EEPROM Memory e appare come nella figura in alto. Il contenuto della memoria non può essere modificato da questa finestra, ma grazie a un'altra opzione del menù è possibile eseguire delle modifiche.

## Absolute Listing

Se selezioniamo questa finestra appare il file del nostro progetto con estensione ".lst". Durante la compilazione si crea un file con lo

*Finestra dello stack.*

stesso nome del nostro programma sorgente ma con questa estensione e selezionando la finestra Absolute Listing possiamo editare il file. In esso troviamo il codice sorgente in modo assoluto e il codice oggetto generato, così come possiamo vedere nella figura.

Al termine di questo file troveremo l'informazione delle etichette utilizzate nel programma, su quale linea si trovano, la memoria utilizzata, la memoria libera e gli errori, warning e messaggi generati dal compilatore.

## Stack

Grazie a questa opzione potremo visualizzare il contenuto dello stack. Il contenuto può essere mostrato con o senza il numero di linea. Sceglieremo il formato di presentazione cliccando sull'icona nella parte superiore sinistra della finestra.

Se il contenuto eccede oltre la dimensione dello stack MPLAB lo indica mostrando il messaggio "underflow".

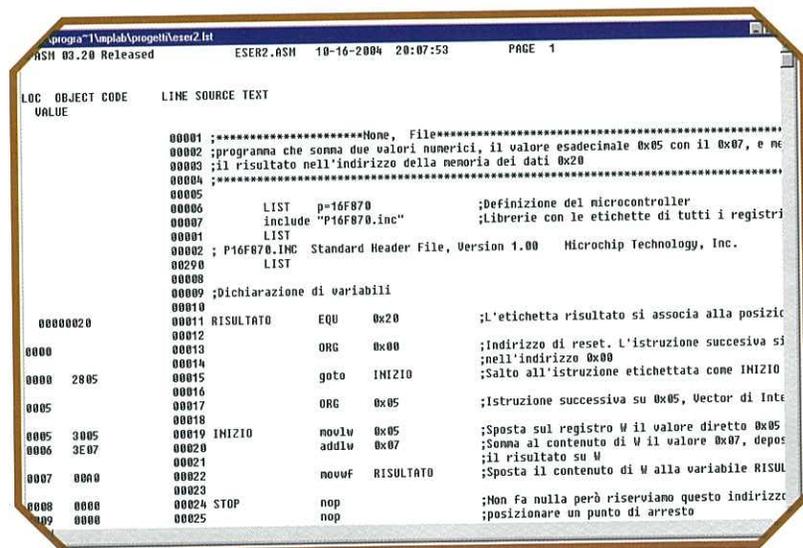
## Conclusioni

La simulazione non avrebbe senso se non si potesse utilizzare il funzionamento interno del PIC e, per questa ragione, è necessario vedere tutte le finestre che ci offre MPLAB. Conoscendo queste e imparando bene la gestione del simulatore sarà più facile mettere a punto il programma per una sua successiva scrittura sul microcontroller. Ma questo lo vedremo nei prossimi capitoli.

```
\progra~1\mplab\progetti\eser2.lst
ASM 03.20 Released          ESER2.ASM   10-16-2004  20:07:53       PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00001 ;*********************Nome,  File*************************************
                      00002 ;programma che somma due valori numerici, il valore esadecimale 0x05 con il 0x07, e me
                      00003 ;il risultato nell'indirizzo della memoria dei dati 0x20
                      00004 ;***********************************************************************
                      00005
                      00006         LIST    p=16F870                 ;Definizione del microcontroller
                      00007         include "P16F870.inc"            ;Librerie con le etichette di tutti i registri
                      00001         LIST
                      00002 ; P16F870.INC  Standard Header File, Version 1.00    Microchip Technology, Inc.
                      00290         LIST
                      00008
                      00009 ;Dichiarazione di variabili
                      00010
  00000020            00011 RISULTATO     EQU     0x20             ;L'etichetta risultato si associa alla posizic
                      00012
0000                  00013               ORG     0x00             ;Indirizzo di reset. L'istruzione succesiva si
                      00014                                        ;nell'indirizzo 0x00
0000   2805           00015               goto    INIZIO           ;Salto all'istruzione etichettata come INIZIO
                      00016
0005                  00017               ORG     0x05             ;Istruzione successiva su 0x05, Vector di Inte
                      00018
0005   3005           00019 INIZIO        movlw   0x05             ;Sposta sul registro W il valore diretto 0x05
0006   3E07           00020               addlw   0x07             ;Somma al contenuto di W il valore 0x07, depos
                      00021                                        ;il risultato su W
0007   00A0           00022               movwf   RISULTATO        ;Sposta il contenuto di W alla variabile RISUL
                      00023
0008   0000           00024 STOP          nop                      ;Non fa nulla però riserviamo questo indirizzc
 009   0000           00025               nop                      ;posizionare un punto di arresto
```

*File "eser2.lst" visibile selezionando Absolute Listing.*

# Gli interrupt

***Un interrupt consiste nell'interruzione del programma in corso per realizzare una determinata routine che risolve la causa che ha provocato l'interrupt. Non tutti i microcontroller dispongono di questo dispositivo, però man mano che si capisce il concetto ci si rende conto di quanto sia importante e pratico. Verificheremo che nella maggior parte delle applicazioni sviluppate con il PIC l'utilizzo degli interrupt risulta fondamentale.***

## Generalità

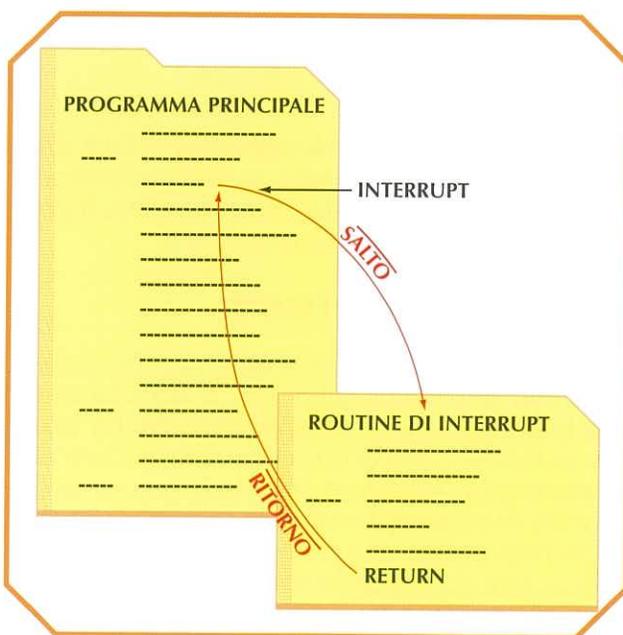
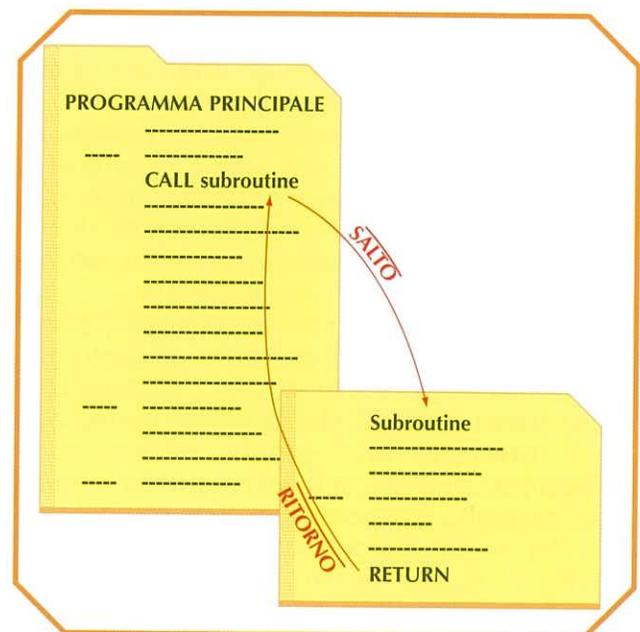
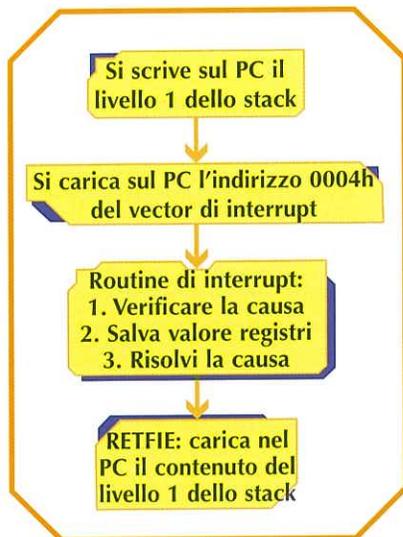
Un interrupt è un'interruzione nel flusso di controllo del programma principale motivata da una causa speciale.

Quando si genera questo evento speciale, se il processore accetta l'interrupt, salva il valore del contatore di programma (PC) nello stack, e lo carica con il Vector di Interrupt a cui corrisponde l'indirizzo 0004h della memoria delle istruzioni. In questa posizione si trova la prima istruzione di una routine dedicata all'interrupt e alla causa che l'ha provocato. Più avanti vedremo che questa istruzione è un'istruzione di salto.

Al termine della routine di interrupt si ritorna al programma principale, nel punto in cui lo si è abbandonato.

Un interrupt è come una chiamata a subroutine, originato da una causa diversa da un'istruzione di tipo CALL. In entrambi i casi si salva il valore del PC sullo stack, si esegue la routine associata e, terminata l'esecuzione, si ritorna al programma principale, però gli interrupt sono delle rotture asincrone del flusso di programma (non si conosce il momento in cui si generano), mentre le chiamate a subroutine sono sincrone (si sa quando si verificheranno). Nelle figure possiamo vedere le similitudini tra un interrupt e una chiamata a subroutine.

## Applicazione dell'interrupt

La ragione per cui affermiamo che l'utilizzo degli interrupt risulta fondamentale per un programmatore è che, grazie a questi, è possibile far fronte in modo immediato agli avvenimenti del mondo esterno. Una delle possibili cause, come vedremo successivamente, di ge-

*Quando si genera un interrupt si esegue la routine di interrupt e si ritorna al programma.*

*In una chiamata a subroutine lavoriamo in modo sincrono.*

*Sequenza di operazioni che comporta un interrupt.*

*Sensori. Quando cambiano il loro stato si produce un interrupt.*

*Diverse tastiere e pulsanti che si gestiranno utilizzando interrupt.*

nerazione di interrupt, è l'applicazione di un livello o di un particolare fronte su uno dei pin del microcontroller. Qualsiasi applicazione che utilizzi un sensore digitale come il sensore di pioggia di un'automobile, il sensore di presenza in un sistema di allarme, il sensore di fumo in un sistema antincendio, un sensore finecorsa, ecc., utilizzerà gli interrupt per comunicare al microprocessore lo stato in cui si trova. Le applicazioni che utilizzano una tastiera o dei pulsanti, come il telecomando di un garage, il telefono cellulare, una cassa del supermercato, ecc., provocano anch'essi, ogni volta che si preme un pulsante, un interrupt che il processore si incarica di gestire.

In qualsiasi progetto potremmo trovare la necessità di interagire con elementi esterni e molti di essi saranno gestiti tramite interrupt. Nelle figure successive sono presentati alcuni esempi di elementi esterni più comuni con cui può lavorare il nostro microcontroller.

Gli elementi esterni sono molto utili per capire le diverse applicazioni che si possono dare agli interrupt, esistono, però, anche delle cause interne al PIC che possono provocare degli interrupt. Immaginate, ad esempio, un'applicazione che, a un certo punto, debba realizzare una funzione specifica, come il calcolo del tempo di possesso del pallone in una partita di pallacanestro. Quando termina il conteggio è necessario attivare immediatamente un avviso acustico. Il PIC, grazie ai suoi temporizzatori interni, realizzerà il conteggio e quando questo termina provocherà un interrupt. Nella routine di interrupt si farà suonare l'avvisatore acustico e si reinizializzeranno i valori del conteggio, indicando che il possesso di palla ha cambiato squadra.

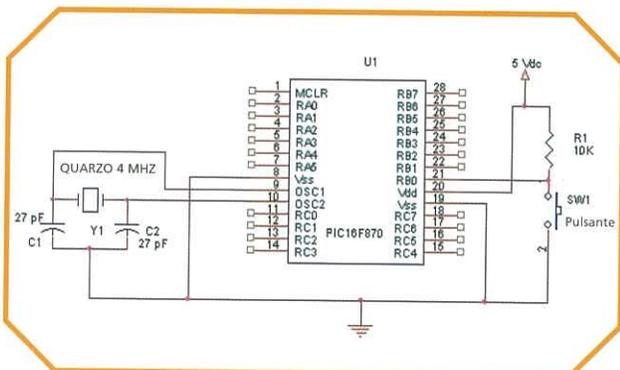
## Il Vector di Interrupt

Qualunque sia la causa che provochi l'interrupt, nei microcontroller PIC, si utilizza sempre lo stesso Vector di Interrupt che ha assegnata la posizione di memoria 0004h. Quando si genera un interrupt, il primo lavoro della routine di interrupt è identificare la causa che lo ha provocato, per poter quindi saltare alla subroutine corrispondente, dato che ci saranno tante subroutine quante le possibili cause di interrupt.

Se un programma inizia all'indirizzo 0000h, che corrisponde all'indirizzo del Vector di Re-

Esempio di interrupt interni nel PIC.

set, e abbiamo detto che il Vector di Interrupt occupa l'indirizzo 0004h, cosa deve fare il nostro programma principale per evitare di sovrascrivere il Vector di Interrupt? Nel nostro programma occorrerà inserire nella posizione 0000h un'istruzione di salto (GOTO) all'indirizzo 0005h, in modo da schivare e non inquinare il dato presente nel Vector di Interrupt. Anche all'indirizzo 0004h dovrà esserci un'istruzione di salto a un altro indirizzo dove ci sia uno spazio libero adatto a ospitare la routine di interrupt. Normalmente si iniziano tutti i programmi nello stesso modo, così come mostrato nell'esempio della figura.

Tipico modo con cui inizieremo un programma.

Esempio di interrupt per RB0/INT.

## Cause di interrupt

La famiglia PIC16F870/1 accetta 14 possibili cause che possono provocare un interrupt. Solo queste richieste avranno il permesso del microcontroller e solo queste cause saranno riconosciute dal PIC come generatrici di interrupt:

**Interrupt per cause esterne:**

1ª. Attivazione del pin RB0/INT: è la più utilizzata e la più semplice. Per abilitare questo interrupt si imposta a 1 il bit INTE (INTCON<4>). In questo modo, quando l'elemento esterno collegato al pin RB0 genera un fronte di salita – bit INTEDG =(OPTION_REG<6>) impostato a 1– o un fronte di discesa (bit INTEDG posto a 0) si attiva il bit INTF (INTCON<1>), si rileva che si è generato un interrupt. Se il bit di abilitazione globale GIE (INTCON<7>) è abilitato, il processore salterà a eseguire la routine di servizio all'interrupt (ISR).

Nella figura si può vedere un esempio di come possiamo generare un interrupt di questo tipo. Quando si attiva il pulsante si produrrà un fronte di discesa che provocherà un interrupt, sempre se i registri sono stati configurati per accettare un interrupt di questo tipo. Immaginate un microrobot con un sensore finecorsa sulla parte frontale. Quando il microrobot incontra un ostacolo sulla sua traiettoria, si chiederà il contatto del finecorsa e produrrà un fronte di discesa in RB0.

2ª. Cambio di stato di uno dei pin RB7:4 della porta B: quando si produce un cambio di stato su qualcuno dei pin di ingresso RB7, RB6, RB5 o RB4, il flag RBIF si imposterà a 1 (bit 0 del registro INTCON). Questi interrupt possono abilitare o disabilitare attivando o ponendo a 0 il bit RB1E (INTCON<4>) di abilitazione. La principale applicazione per cui questo tipo di interrupt è utilizzato è il possibile collegamento di sensori alla porta. Ad esempio, se abbiamo collegato quattro sensori di presenza per un'applicazione di sicurezza, nel momento in cui uno di essi si atti-

| | BIT REGISTRO<POS> | Descrizione |
|---|---|---|
| RB0/INT | PORTB<0> | Pin dove si collegherà l'elemento esterno |
| INTE | INTCON<4> | Bit di abilitazione di questo tipo di interrupt |
| GIE | INTCON<7> | Bit di abilitazione di interrupt (permette che si possano verificare) |
| INTEDG | OPTION_REG<6> | Selezioniamo quale fronte sarà la causa di interrupt (a 1 quello di salita, a 0 quello di discesa) |
| INTF | INTCON<1> | Flag che indica che si è verificato questo tipo di interrupt |

*Bit che intervengono quando si produce un interrupt tramite RB0.*

va, cambia lo stato e attiverà il flag, e se gli interrupt saranno stati abilitati, verrà eseguita la routine di gestione dell'interrupt.

Per gli interrupt generati da elementi esterni, come per i precedenti, la latenza dell'interrupt durerà tre o quattro cicli di istruzione. Il tempo esatto di latenza dipenderà dal momento in cui si produce l'interrupt.

**Interrupt per cause interne:**

3ª. Termine di un'operazione di scrittura nella EEPROM:

In questo caso, quando il processo interno di scrittura nella memoria di programma EEPROM termina, il bit flag EEIF (PIR2<4>) si attiva e se il bit di abilitazione dell'interrupt EEIE (PIE2<4>) è attivato, si produce l'interrupt.

Sensore 1 Sensore 2 Sensore 3 Sensore 4

SENSORI DI PRESENZA

U1

MCLR RA0 RA1 RA2 RA3 RA4 RA5 Vss OSC1 OSC2 RC0 RC1 RC2 RC3

RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0 Vdd Vss RC7 RC6 RC5 RC4

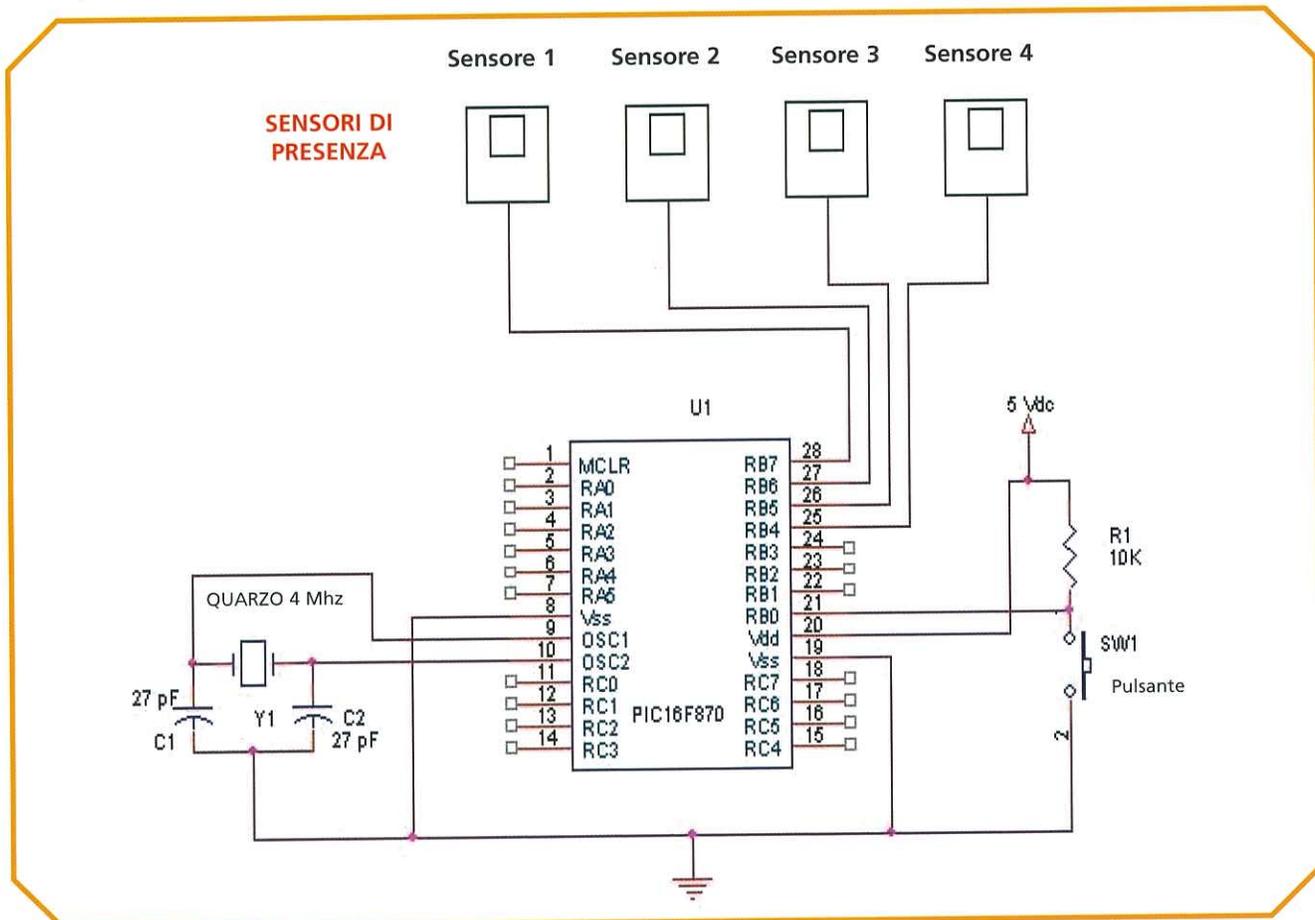
PIC16F870

5 Vdc

R1 10K

SW1 Pulsante

QUARZO 4 Mhz

27 pF C1 Y1 C2 27 pF

*Esempio di un sistema di allarmi che utilizza gli interrupt tramite la porta B.*